REMARKS

Claims 1-3, 6, 8-13, 17-18, 23-25, 28, 30-35, 39-40, 61-63, 66, 68-73, and 77-78 are pending in the present application. Claims 1, 23, and 61 are the independent claims. Claims 1-3, 6, 8-13, 17-18, 23-25, 28, 30-35, 39-40, 61-63, 66, 68-73, and 77-78 stand rejected.

Claim Rejections - 35 USC § 112

Claims 1-3, 6, 8-13, 17-18, 23-25, 28, 30-35, 39-40, 61-63, 66, 68-73, and 77-78 stand rejected under 35 USC § 112, ¶2, as being indefinite. Official Action, at 4. Specifically, the Office states:

> Claims 1-3, 6, 8-13, 17-18 recite a method…However, the claims direct to another scope; it appears an apparatus which is merely for describing a type of programming language specification rather than an act of the method:
>
> …
>
> Claims 6- 15, 17- 19 are indefinite because the claims are apparatus of programming specification.
>
> Claims 23-25,28, 30-35, 39-40 recite a scope of a computer readable storage medium to store executable instructions. However, the claimed subject, "instructions" is indefinite because it does not point out this particularly "instruction" and what it does.
>
> …
>
> Claims 61-63, 66, 68-73,77-78 recite a method generating an object using a compiler…The Claims attempt to use compiler but it does not show compiling. Its claiming in term of enablement is in question. The functionality in the claims is ambiguous, and the claimed subject matters in the claims are unclear.

Official Action, at 4-7.

Applicants have amended **claim 1** to recite, in part,

> after implementing in the class the first and second explicit interfaces, compiling the class into computer-executable instructions along with a call in the class to the first explicit interface member and a call in the class to the second explicit interface member, the call comprising an identifier of an explicit interface, and a corresponding identifier of an explicit interface member;

> in response to processing a first instruction of the instructions corresponding to the call to the first explicit interface member, executing the call with the first explicit interface member to produce a first result;
>
> storing the first result in a computer readable storage medium;
>
> in response to processing a second instruction of the instructions corresponding to the call to the second explicit interface member, executing the call with the second explicit interface member to produce a second result; and
>
> storing the second result in the computer readable storage medium.

Support for this language is found in the original disclosure in at least the Specification, at 18-19 and FIG. 8F. Applicants request that the Office reconsider the rejection of claim 1 at least in light of the claim amendment, as well as the rejections of dependent **claims 2-3, 6, 7-13, or 17-18**.

Applicants have amended independent **claims 23 and 61** in a similar fashion as they have amended claim 1. Applicants request that the Office reconsider the rejections of claim 23 and 61 at least in light of the claim amendments, as well as the rejections of dependent **claims 24-25, 28, 30-35, 39-40, 62-63, 66, 68-73, and 77-78.**

Claim Rejections - 35 USC § 103

Claims 1-3, 6, 8-13, 17-18, 23-25, 28, 30-35, 39-40, 61-63, 66, 68-73, and 77-78 stand rejected under 35 USC § 103 as being unpatentable over Joseph Bergin, "Multiple Inheritance in Java" in view of Shindich, "A Few Words About Python Interfaces." Official Action, at 7-8.

Applicants have amended **claim 1 to** recite, in part,

> ***implementing in a class a first explicit interface having a first explicit interface member by explicitly specifying the relationship between the class and the first explicit interface member***, the first explicit interface member being excluded from a public interface of the class;
>
> ***implementing in the class a second explicit interface having a second explicit interface member***, the second explicit interface member having the same signature as the first explicit interface member, ***a signature comprising a member name, a number of***

> ***one or more parameters, and a type for each of the number of one or more parameters***;

(Emphasis added). Support for this amendment is found in the original disclosure in at least the Specification, at 18, lines 29-31.

The Office states that Bergin teaches the similar language of "implementing in the class a second explicit interface member, the second explicit interface member having the same signature as the first explicit interface member" in the following manner:

> See p. 4, the member "parent child". Either Parent child or Other Child has the same signature child member to its parent.

Official Action, at 8. Even if Bergin teaches this similar language, applicants submit that Bergin fails to teach this excerpted language of claim 1. The Office's citation to Bergin teaches:

```
Interface OtherInterface
{       void whatever();
}

class OtherChild extends Other implements OtherInterface
{       public OtherChild (int value){ super(value); }
}

Class ParentChild extends Parent implements OtherInterface
{       public ParentChild(...) { child = new OtherChild(...); ... }

        public void whatever() { child.whatever(); }

        private final OtherInterface child;
}
```

Bergin, at 4. This citation to Bergin shows one interface – OtherInterface, and it shows that class ParentChild implements one interface – OtherInterface (and also extends one class – Parent). However, claim 1 recites both "implementing in a class a first explicit interface having a first explicit interface member" and "implementing in the class a second explicit interface having a second explicit interface member." That is, Bergin teaches implementing only one interface, and claim 1 recites implementing both a first explicit interface and a second explicit interface.

Furthermore, claim 1 recites that this implementation of interfaces involves,

> implementing in a class a first explicit interface having a first explicit interface member…[and] implementing in the class a second explicit interface having a second explicit interface

> member, ***the second explicit interface member having the same signature as the first explicit interface member***, a signature comprising a member name, a number of one or more parameters, and a type for each of the number of one or more parameters.

(Emphasis added). Bergin teaches implementing two methods in ParentChild: ParentChild(…) and whatever(). Neither of these have the same signature as claimed, which requires that a signature comprise "a member name, a number of one or more parameters, and a type for each of the number of one or more parameters." As can be seen, ParentChild(…) and whatever() have different names, and "…" is commonly used in the art to denote at least one parameter, while whatever has no parameters, so they do not have the same number of parameters, either.

The Office also cites to Shindich in rejecting claim 1. However, Shindich fails to cure the deficiencies of Bergin. Shindich does appear to teach explicitly implementing two interfaces:

```
interface Musician:
     def playViolin ():
          """""""
interface Philosopher:
     def thinkHard ():
          """""""
interface Programmer:
     def hackCode ():
          """""""
class Son1 (Mother, Father) implements
Musician, Philosopher:
```

Shindich. However, Shindich does not teach that the implementation of multiple interfaces involves interface members having the same signature. To wit, the above excerpt from Shindich contains the only three interfaces that Shindich teaches: Musician, Philosopher, and Programmer. None of these three interfaces has a member that shares a signature with any other interface's member. The only member of the Musician interface described is playViolin(); the only member of the Philosopher interface described is thinkHard(); and the only member of the Programmer interface described is hackCode(). These three members – playViolin(), thinkHard(), and hackCode() – have different signatures because claim 1 recites that a signature comprises a member name and these three members have different names.

For at least these reasons, applicants submit that the combination of Bergin and Shindich fails to render obvious the invention described in independent claim 1, or in dependent **claims 2-3, 6, 7-13, or 17-18**. Applicants request that the Office reconsider the rejections.

Applicants have amended independent **claims 23 and 61** in a similar fashion as they have amended claim 1. Applicants submit that the combination of Bergin and Shindich fails to render obvious the inventions described in independent claims 23 or 61 for at least the reason that the combination of Bergin and Shindich fails to render obvious the invention described in independent claim 1. Inasmuch as **claims 24-25, 28, 30-35, 39-40, 62-63, 66, 68-73, and 77-78** depend from claims 23 or 61, applicants submit that the combination of Bergin and Shindich fails to render the inventions described in those claims. Applicants request that the Office reconsider the rejections.

Applicants have also amended claim 61 to recite, "a processor; and
a memory communicatively coupled to the processor when the system is operational, the memory bearing processor-executable instructions that, when executed on the processor, cause…" Support for this amendment is found in the original disclosure in at least the Specification, at 21.

CONCLUSION

In the view of the foregoing amendments and remarks, Applicants respectfully submit that the present application is in condition for allowance. Reconsideration of the application and an early Notice of Allowance are respectfully requested. In the event that the Examiner cannot allow the application for any reason, the Examiner is encouraged to contact Applicants' representative.

Date: April 25, 2011

/Peter Trahms-Neudorfer/
Peter Trahms-Neudorfer
Registration No. 59,282

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439